# **وثيقة تحليل نظام إدارة العيادات الأولية**

## **1. المقدمة**

### **1.1. الهدف من الوثيقة**

* تحديد متطلبات النظام من حيث الوظائف والواجهات والأداء.
* تحديد حدود النظام والتفاعل مع الأنظمة أو الجهات الخارجية.
* تقديم رؤية واضحة لبناء الحل باستخدام التقنيات: C# مع ASP.NET Core 9، MVC للباك إند، React للفرونت إند، SQL Server لقواعد البيانات.
* تقسيم الحل إلى ثلاث مشاريع داخل السليوشن:
  1. **مشروع الكيانات وقواعد البيانات (Entities & Data Access):** يحتوي على النماذج (Models) وتعريف العلاقات وقاعدة البيانات باستخدام ORM مثل Entity Framework Core.
  2. **مشروع إجراءات العيادة (Clinic Operations):** يحتوي على الخدمات والعمليات المتعلقة بإدارة ملفات المرضى، جداول المواعيد، وتفاعل الإداريين والأخصائيين.
  3. **مشروع إدارة التقييمات (Evaluations Management):** يحتوي على إنشاء وإدارة الاستبيانات الثابتة، حساب الدرجات وإصدار النتائج النهائية.

### **1.2. نطاق النظام**

* تطبيق لإدارة العيادات الأولية يتضمن إدارة المرضى، المواعيد، الملفات الطبية، وتقييم أداء الأخصائيين، بالإضافة إلى إدارة التقييمات الثابتة.
* دعم عدة أدوار للمستخدمين مع صلاحيات مختلفة.
* توفير واجهة تفاعلية للمستخدمين باستخدام React للتعامل مع العمليات بشكل سلس وفعال.

## **2. وصف النظام**

### **2.1. نظرة عامة على الوظائف الأساسية**

* **إدارة المستخدمين والأدوار:** تسجيل الدخول وتفويض الصلاحيات للمستخدمين (سوبر أدمن، أدمن، مدير عيادة، أخصائي، جهة عمل).
* **إدارة ملفات المرضى:** فتح ملفات جديدة عند زيارة المريض أو إحالته، متابعة الملف الطبي، تسجيل الأحداث والتقييمات.
* **جدولة المواعيد:** تحديد المواعيد بين المرضى والأخصائيين، تعديل وتحويل المواعيد.
* **إدارة التقييمات:** إنشاء استبيانات ثابتة مسبقاً، تعيين الدرجات لكل إجابة، حساب النتيجة النهائية وتحديد مستوى استجابة المريض للعلاج.
* **إدارة العيادات والإدارات:** مراقبة أداء العيادات من قبل مديري العيادات والإداريين ومتابعة سير العلاج.
* **متابعة جهة العمل:** تمكين جهة العمل من متابعة حالة المريض والمواعيد والتقارير بشكل سطحي دون إمكانية التعديل.

### **2.2. التقنيات المستخدمة**

* **الباك إند:** ASP.NET Core 9 باستخدام MVC، RESTful API.
* **الفرونت إند:** React مع مكتبات مثل Redux أو Context API لإدارة الحالة.
* **قواعد البيانات:** SQL Server مع استخدام Entity Framework Core في مشروع الكيانات.
* **التقسيم المعماري:** تقسيم الحل إلى ثلاث مشاريع (Entities & Data Access، Clinic Operations، Evaluations Management) لدعم الفصل بين المسؤوليات وتحسين الصيانة.

## **3. تحليل المتطلبات الوظيفية**

### **3.1. المستخدمون والأدوار وصلاحياتهم**

#### **3.1.1. سوبر أدمن**

* يمتلك جميع الصلاحيات على النظام.
* إدارة جميع المستخدمين والأدوار.
* التحكم الكامل في إعدادات النظام والتكوينات.

#### **3.1.2. أدمن إدارة العيادات**

* إدارة جميع العيادات.
* إضافة وتعديل المستخدمين (الإداريين، مديري العيادات، الأخصائيين).
* متابعة ملفات المرضى.
* تقييم أداء الأخصائيين.
* إدارة التقييمات (الإضافة، التعديل، تعديل الدرجات قبل اعتمادها).
* تحويل المرضى لأقسام أخرى وإنهاء العلاج وإقفال الملف.

#### **3.1.3. مديري العيادات**

* إدارة العيادات التابعة لقسمهم.
* متابعة ملفات المرضى الخاصة بعياداتهم.
* متابعة أداء الأخصائيين في العيادات التابعة لهم.
* تحويل المرضى لأقسام أخرى وإنهاء العلاج وإقفال الملف.

#### **3.1.4. الأخصائيين**

* إدارة ملفات المرضى.
* تسجيل مواعيد المرضى وبناء الملف الطبي.
* تسجيل الأحداث والتقييمات النصية والطبية.
* رفع المرفقات الإضافية.
* إنهاء العلاج وإقفال الملف.
* إمكانية تحويل ملف المريض لأخصائي آخر عند الحاجة.

#### **3.1.5. جهة عمل المريض**

* متابعة حالة المريض بشكل سطحي.
* الاطلاع على مواعيد المريض وحالة الملف الطبي (تحت العلاج، بانتظار موعد، ملف مغلق).
* الحصول على التقارير النهائية والتوصيات دون صلاحية تعديل.

### **3.2. تدفق العمليات الرئيسية (Use Cases)**

#### **3.2.1. فتح ملف جديد للمريض**

* **المستخدم:** جهة عمل/مريض ذاتياً.
* **الخطوات:**
  1. تقديم البيانات الأساسية (رقم الوظيفة، جهة العمل، تاريخ فتح الملف، نوع الملف: إحالة أو زيارة ذاتية).
  2. إنشاء الملف وربطه بالبيانات المدخلة.
  3. إبلاغ مدير العيادة لبدء الإجراءات التالية.

#### **3.2.2. تحويل الملف إلى أخصائي وتحديد موعد**

* **المستخدم:** مدير العيادة.
* **الخطوات:**
  1. مراجعة الملف.
  2. تحويل الملف إلى أخصائي محدد.
  3. تحديد موعد لزيارة الأخصائي.
  4. متابعة التطورات على الملف وتسجيل التحويلات عند الحاجة.

#### **3.2.3. بناء الملف الطبي وتقييم المريض**

* **المستخدم:** أخصائي.
* **الخطوات:**
  1. تحديد موعد للمريض.
  2. استخدام نظام التقييمات لإنشاء التقييمات المدخلة مسبقاً.
  3. تسجيل النتيجة النهائية بناءً على الإجابات.
  4. كتابة تقييم نصي، الرأي الطبي والملاحظات.
  5. حفظ البيانات في سجل الملف الطبي.
  6. اتخاذ قرار: تحديد موعد جديد، إنهاء العلاج وإقفال الملف أو تحويله لجهة أخرى.

#### **3.2.4. متابعة جهة العمل**

* **المستخدم:** جهة عمل المريض.
* **الخطوات:**
  1. الدخول إلى النظام باستخدام صلاحيات محدودة.
  2. الاطلاع على حالة المريض ومواعيد الزيارات.
  3. مشاهدة التقرير النهائي والتوصيات عند إقفال الملف.
  4. بعد فترة محددة من إقفال الملف، يُخفى الملف من القائمة دون حذفه.

## **4. المتطلبات غير الوظيفية**

### **4.1. الأداء**

* يجب أن يكون النظام قادرًا على التعامل مع عدد كبير من المستخدمين المتزامنين.
* تحسين استجابة النظام خاصة في عمليات البحث والاستعلام عن الملفات.

### **4.2. الأمان**

* تنفيذ نظام تحقق قوي (Authentication) وتفويض دقيق (Authorization) لكل دور.
* تأمين نقل البيانات باستخدام بروتوكولات HTTPS.
* حماية قواعد البيانات من الهجمات (SQL Injection، وغيرها).

### **4.3. القابلية للتوسع**

* تصميم معماري يدعم إضافة وحدات جديدة في المستقبل (مثلاً، وحدات تقارير متقدمة).
* فصل طبقات التطبيق (Data Access, Business Logic, Presentation) لتسهيل التوسع والتعديل.

### **4.4. القابلية للصيانة**

* استخدام مبادئ التصميم الجيد (SOLID principles) ونمط Repository Pattern لتسهيل الصيانة.
* توثيق الكود واستخدام التعليقات التوضيحية لتسهيل عملية الفهم والتعديل.

### **4.5. تجربة المستخدم**

* واجهة تفاعلية وسهلة الاستخدام باستخدام React.
* تصميم استجابي (Responsive Design) لدعم مختلف الأجهزة.

## **5. المعمارية والتقسيم الداخلي للمشاريع**

### **5.1. مشروع الكيانات وقواعد البيانات (Entities & Data Access)**

* تعريف النماذج (Models) مثل:
  + **User** مع الخصائص (اسم المستخدم، كلمة المرور، الدور، …).
  + **Clinic** يمثل بيانات العيادة.
  + **PatientFile** يحتوي على البيانات المتعلقة بملف المريض (الرقم الوظيفي، جهة العمل، تاريخ الزيارة، الحالة، …).
  + **Appointment** لتسجيل المواعيد.
  + **Evaluation** يحتوي على بيانات الاستبيانات والإجابات والدرجات.
* إنشاء العلاقات بين الكيانات باستخدام Entity Framework Core.
* تطبيق Repository Pattern للوصول إلى البيانات.

### **5.2. مشروع إجراءات العيادة (Clinic Operations)**

* يحتوي على:
  + **Business Logic Layer (BLL):** الخدمات التي تتعامل مع عمليات فتح الملفات، تحويل المرضى، جدولة المواعيد، متابعة الإجراءات.
  + **Controllers:** تنفيذ المنطق الخاص بواجهات الـ API التي تخدم الـ React.
  + **Validation & Exception Handling:** التحقق من صحة البيانات والتعامل مع الأخطاء.

### **5.3. مشروع إدارة التقييمات (Evaluations Management)**

* يحتوي على:
  + إدارة إنشاء وتعديل الاستبيانات الثابتة.
  + حساب الدرجات وفقاً للإجابات وتحديد النتيجة النهائية.
  + توفير واجهات API خاصة للتقييمات يمكن استخدامها من قبل وحدة الأخصائيين.
  + إدارة تاريخ التقييمات وتسجيل التغييرات قبل اعتمادها.

## **6. الواجهات ونقاط التكامل (Integration Points)**

### **6.1. بين الباك إند والفرونت إند**

* استخدام RESTful API لتبادل البيانات بين ASP.NET Core MVC وواجهة React.
* تنسيق البيانات (JSON) بين العميل والخادم.

### **6.2. التكامل مع قاعدة البيانات**

* استخدام Entity Framework Core لإجراء عمليات CRUD على قاعدة بيانات SQL Server.
* إدارة الترقيات (Migrations) لتحديث بنية قاعدة البيانات عند الحاجة.

### **6.3. التكامل مع أنظمة أخرى (إن وجد)**

* إمكانية دمج النظام مع أنظمة الجهات الخارجية لمتابعة المرضى أو تقارير إضافية (يتم تحديد ذلك في مراحل لاحقة إذا دعت الحاجة).

## **7. السيناريوهات ودراسات الحالة (Use Case Scenarios)**

### **7.1. السيناريو الأول: إحالة مريض جديد**

1. يقوم جهة العمل بإحالة مريض.
2. يتم فتح ملف جديد في النظام مع البيانات المطلوبة.
3. يقوم مدير العيادة بمراجعة الملف وتحديد الأخصائي المناسب.
4. يتم جدولة موعد للأخصائي وتحديث حالة الملف.
5. يقوم الأخصائي ببناء الملف الطبي وإجراء التقييمات.
6. يتم إقفال الملف عند الانتهاء مع حفظ التقرير النهائي.

### **7.2. السيناريو الثاني: تحويل ملف لمريض قائم**

1. أثناء متابعة ملف مريض تحت العلاج، يرى مدير العيادة حاجة لتحويل الملف.
2. يقوم مدير العيادة بتحويل الملف إلى أخصائي آخر مع تحديث بيانات الموعد.
3. يستمر الأخصائي الجديد في متابعة الملف وتسجيل التقييمات والإجراءات.

## **8. الفرضيات والقيود**

### **8.1. الفرضيات**

* يفترض أن جميع المستخدمين لديهم اتصال إنترنت مستقر للوصول إلى النظام.
* سيتم توفير بيانات اعتماد لكل دور عند البدء.

### **8.2. القيود**

* يجب الالتزام بسياسات الأمان الخاصة بالمؤسسة.
* قد توجد قيود على عمليات التعديل على ملفات المرضى بمجرد إقفالها.
* قيود زمنية على مدة بقاء البيانات الخاصة بجهة العمل في النظام قبل إخفائها.

## **9. متطلبات التقارير والتنبيهات**

### **9.1. التقارير**

* تقارير متابعة أداء العيادات.
* تقارير متابعة أداء الأخصائيين.
* تقارير تقييم المرضى والتوصيات النهائية.

### **9.2. التنبيهات**

* تنبيهات عند وجود مواعيد قادمة أو تغييرات في حالة الملفات.
* تنبيهات إلى المشرفين في حال حدوث أخطاء في إدخال البيانات أو تجاوزات في التقييمات.

## **10. خطة التنفيذ والاختبارات**

### **10.1. مراحل التنفيذ**

* **مرحلة التحليل والتصميم:** إعداد وثيقة التحليل والتصميم المفصل.
* **مرحلة التطوير:** تقسيم الفرق وفقاً للمشاريع الثلاثة والعمل بالتوازي.
* **مرحلة التكامل:** ربط المشاريع معاً واختبار التكامل بين الطبقات.
* **مرحلة الاختبار والتوثيق:** إجراء اختبارات الوحدة (Unit Testing)، الاختبارات التكامليّة (Integration Testing)، وتجربة المستخدم (UAT).

### **10.2. خطة الاختبار**

* إعداد خطط اختبار لكل وحدة من النظام.
* إجراء اختبارات على كل من الوظائف الأساسية والسيناريوهات الحرجة.
* مراجعة الأداء والأمان قبل إطلاق النظام.

## **11. الملاحق**

### **11.1. نماذج البيانات (ER Diagrams)**

* رسم تخطيطي يوضح العلاقات بين الجداول (Patients, Clinics, Appointments, Evaluations, Users، إلخ).

### **11.2. نماذج الواجهات (Wireframes)**

* تصميمات أولية لواجهات React التي ستستخدم لكل من:
  + لوحة التحكم الرئيسية.
  + صفحات إدارة المرضى.
  + صفحات إدارة التقييمات.
  + صفحات التقارير والمتابعة.

### **11.3. جداول الصلاحيات (Access Control Matrices)**

* تحديد دقيق للصلاحيات لكل دور والمزايا المتاحة لكل نوع من المستخدمين.